Two, Four, Six, Eight! Software and Systems – Integrate!

Here I am, in yet another airport, writing a BACKTALK column. The theme this month? Software and systems integration. And, of course, this reminds me of story.

I was lucky—my parents bought me a brand new 1973 Chevy Impala right out of high school[1]. The 1973 Chevy Impala, although a great car, came with a very basic AM/FM radio.

Keeping in touch with the times, I wanted to be "cool" and install a cassette player (contrary to stories, I was not old enough to have wanted an eight-track player). I saved up my dollars, and eventually went to a local auto parts shop and bought an in-dash AM/FM/cassette radio. It came with a complete book of instructions.

Really, how hard could it be? Well, the first instruction was to *remove the old radio*, and provided a helpful hint that perhaps *dropping the dashboard* was the way to go. So, armed with a set of borrowed-from-my-dad screwdrivers, I got to work. Eventually, stuff started to come off and I was able to see the mounting brackets for the old radio. Carefully, I removed it and isolated the wires.

I remember that there were about eight wires, almost all of them black. A few had some color-coded tags, but some did not. There was a socket in the back of the radio, but it had no labeling, and several of the wires had no obvious way to disconnect—they simply disappeared into a hole in the dashboard.

Surely, I thought, the new radio instructions would explain. As I read the instructions, I noted that the new radio had nine (not eight!) wires, each color-coded. I spent the better part of the day trying to match the old and new wires. I blew several fuses. I never could get one speaker to work. I ended up draining the battery, and my dad did not have a battery charger. I ended up rolling the car back, and using dad's car to jump mine.

Sundown came, and I finally gave up. Luckily, I had marked the old wires well, and I was able to reconnect the old radio, reinstall the dash[2], and return the new radio for a "full and prompt refund."

Years later, I bought a car[3] that came with only an AM/FM radio. I asked the dealer how much it cost to install an AM/FM radio with a six-disc CD player. The answer was less than $300, so I said "go for it!" When I was done with the hour-long "signing your life away" paperwork, I asked when I could bring back the car to change out the radio. The salesman laughed, and said "It's done already!" I was told (and then shown) that the new cars had a standard socket for all of their radios, and that the replacement was as simple as "Use special tool to remove old radio, insert new radio." Total time, five minutes!

Standardization. Agreement on interfaces. Planning the interfaces ahead of time. Testing them to make sure they work. Sounds easy, but it's not. You need a 50,000-foot view to see how the system will integrate. Programmers are too low on the food chain to see it. Most designers only understand a single system. True system architects are hard to find.

You see, integration takes time and planning and needs to be done *before* you start building the low-level modules of a system. Many large systems have a chief engineer, but his or her job is to understand the complex technical issues. You still need a chief architect to see how everything needs to fit together, and then design the system to work.

I have always taught that there are four phases of design: architecture, data, interface, and module. Most developers focus on the module design because it's understandable at a low level. Most software engineering processes and tools help with the data design. And, if you are using any type of CASE (Computer-Assisted Software Engineering), the interfaces are controlled. However, the really *big* picture is the architecture. I do not know of any really good tools that automate the overall systems and software integration. Maybe it's time for a CASSI (Computer-Assisted Software and Systems Integration) tool[4] to help create, design, manage, and help identify problem areas.

Which brings me back to my current wait in the airport. I am trying to get home from Seattle to Albuquerque. My first leg to Salt Lake City is already 90 minutes late. Since my layover is only an hour, I am guaranteed to miss my connection. I called the airline to rebook but was told I can't because their computers still show the flight out of Seattle as "on time." I point out that since I am calling (and am still on the ground in Seattle), there is no way it's "on time." The very nice and apologetic flight agent agrees, but says that until somebody officially lists the flight as "delayed," the computer "thinks" things are OK, and I cannot rebook a new connection for free.

Integration is hard. You *need* to plan for it. You *need* a system architect to help with the big picture, and plan the interfaces. You *need* a system architect to visualize, create, and then control the big picture—if you don't have a high-level integration plan, the probability of the software integrating properly is close to zero. And, like the airline, when problems occur, you *need* to recognize them early—and take remedial action. Don't keep listing your program as "on time" when you know there's a problem. It will only get worse.

—David A. Cook, Ph.D.
Principal Member of the Technical Staff,
The AEgis Technologies Group, Inc.
dcook@aegistg.com

Notes

1. Let me point out that I had a choice: go to college car-less, or get a car and go to college locally. I picked the car. I joined the Air Force within a year, so I obviously made the right choice.
2. Although, truth be told, until I got rid of the car in 1986, the speedometer cable rattled and occasionally the fuse to the dash light blew out. I am sure this was a normal occurrence, and had nothing to do with my reinstallation.
3. OK, it *was* a minivan, but driving a minivan *does not* automatically label you as "middle-aged."
4. And I can find no reference to this on the Internet, so if this becomes popular in the future, I've coined a new term!

Can You BACKTALK?

Here is your chance to make your point without your boss censoring your writing. In addition to accepting articles that relate to software engineering for publication in CROSSTALK, we also accept articles for the BACKTALK column. These articles should provide a concise, clever, humorous, and insightful perspective on the software engineering profession or industry or a portion of it. Your BACKTALK article should be entertaining and clever or original in concept, design, or delivery, and should not exceed 750 words.

For more information on how to submit your BACKTALK article, go to <www.stsc.hill.af.mil>.